On an optimal randomized acceptor for graph nonisomorphism*

Edward A. Hirsch[†,‡] Dmitry Itsykson[†]

November 9, 2011

Abstract

An acceptor for a language L is an algorithm that accepts every input in L and does not stop on every input not in L. An acceptor Opt for a language L is called *optimal* if for every acceptor A for this language there exists polynomial p_A such that for every $x \in L$, the running time $\mathrm{time}_{\mathrm{Opt}}(x)$ of Opt on x is bounded by $p_A(\mathrm{time}_A(x) + |x|)$ for every $x \in L$. (Note that the comparison of the running time is done pointwise, i.e., for every word of the language.)

The existence of optimal acceptors is an important open question equivalent to the existence of p-optimal proof systems for many important languages [KP89, Sad99, Mes99]. Yet no nontrivial examples of languages in $\mathbf{NP} \cup \mathbf{co\text{-}NP}$ having optimal acceptors are known.

In this short note we construct a randomized acceptor for graph nonisomorphism that is optimal up to permutations of the vertices of the input graphs, i.e., its running time on a pair of graphs (G_1, G_2) is at most polynomially larger than the maximum (in fact, even the median) of the running time of any other acceptor taken over all permuted pairs $(\pi_1(G_1), \pi_2(G_2))$. One possible motivation is the (pointwise) optimality in the class of acceptors based on graph invariants where the time needed to compute an invariant does not depend much on the representation of the input pair of nonisomorphic graphs.

In fact, our acceptor remains optimal even if the running time is compared to the *average-case* running time over all permuted pairs. We introduce a natural notion of average-case optimality (not depending on the language of graph nonisomorphism) and show that our acceptor remains average-case optimal for any probability distribution on the inputs that respects permutations of vertices.

Keywords: optimal algorithm, graph isomorphism.

1 Introduction

Optimal algorithms. While most complexity theorists believe in the $\mathbf{P} \neq \mathbf{NP}$ conjecture, the existence of the fastest algorithm for any problem in $\mathbf{NP} \setminus \mathbf{P}$ is an intruguing open question. Here by "the fastest" we mean the minimum possible running time for *every* possible input. Formally, an *optimal* algorithm runs in time bounded by a polynomial of the time of any other algorithm (the

*Supported in part by the grants NSh-5282.2010.1 and MK-4089.2010.1 from the President of RF, by the Programme of Fundamental Research of RAS, and by RFBR grants 11-01-00760 and 11-01-12135-ofi-m-2011. The second author is also supported by Rokhlin Fellowship.

[†]Steklov Institute of Mathematics at St.Petersburg, Russian Academy of Sciences, 27 Fontanka, St.Petersburg, 191023, Russia. Phone: +78125714392. Fax: +78123105377. Email: {hirsch,dmitrits}@pdmi.ras.ru, Web: http://logic.pdmi. ras.ru/{~hirsch,~dmitrits}

[‡]Corresponding author.

polynomial is specific for the "simulated" algorithm). Many years ago Levin presented an optimal algorithm [Lev73] for **NP** *search* problems, but it does not translate to *decision* problems, which can be possibly solved more efficiently for some inputs.

Solving a decision problem L can be "split" into two complementary tasks: to give the answer "yes" on the positive instances ($x \in L$) and diverge (do not stop) on the negative ones ($x \notin L$), and to give the answer "no" on the negative instances and diverge on the positive ones. The running time matters on those instances where the algorithm stops. A semi-decision procedure that performs the first task as fast as any other such procedure is called an *optimal acceptor* for L. Optimal acceptors were introduced in [KP89] and studied in connection to p-optimal proof systems (see, e.g., a survey [Hir10]). It was proved that for the language of Boolean tautologies, an optimal acceptor exists if and only if a p-optimal proof system exists; Messner generalized this result to paddable languages [Mes99]. While it is easy to see that Messner's proof goes for randomized acceptors as well, it does not apply to computations with advice (where optimal proof systems exist [CK07], but no optimal acceptors are known), heuristic computations (where optimal acceptors do exist [HINS11], but no optimal proof systems are known), or the restricted notion of optimality used in this paper. Recently, it was shown that the existence of optimal acceptors is equivalent for all **co-NP**-complete languages [CFM11].

Schnorr [Sch76] defines the notion of worst-case optimal acceptor. An acceptor B is worst-case optimal for a language L if for every acceptor A for this language there exists a polynomial p_A such that $\mathrm{time}_B(x) \leq p_A(\max_{|y|\leq|x|} \mathrm{time}_A(y) + |x|)$ for every $x \in L$. Using Levin's optimal algorithm for **NP**-search problems Schnorr shows that every downward selfreducible language L in **NP** (e.g. SAT) has a worst-case optimal acceptor.

To the date, no optimal acceptors are known for languages in $(\mathbf{co\text{-}NP} \cup \mathbf{NP}) \setminus \mathbf{P}$. The same applies to *randomized* acceptors in $(\mathbf{co\text{-}NP} \cup \mathbf{NP}) \setminus \mathbf{BPP}$, i.e., the procedures that can have either one- or two-sided bounded probability of error. The only optimal acceptors for **co-NP**-languages are *heuristic* randomized acceptors, i.e., acceptors with unbounded probability of error for a small fraction of the inputs [HIMS11].

The only example of an (artificial) language beyond **P** having an optimal acceptor is a $\mathbf{Time}(t(n))$-immune set in $\mathbf{Time}(t(n)n^3 \log t(n))$ that simply does not have acceptors that have better-than-$t(n)$ running time for any infinite subset [Mes00]. Chen, Flum and Müller [CFM11] generalize this result from optimal acceptors to optimal algorithms.

Graph (non)isomorphism and optimality up to permutations of vertices. The problem of graph isomorphism is a natural thoroughly studied problem in **NP**. While there is a fast algorithm for it working for almost all instances according to the "uniform" distribution [BK79], the problem is not known to be in **BPP** and it is still possible that there are probability distributions that make it hard on the average.

We study acceptors for graph nonisomorphism, i.e., algorithms that give (presumably fast) answer for nonisomorphic graphs and do not stop on isomorphic ones (the latter can be, of course, compensated by running a brute-force search algorithm in parallel). Most algorithms for graph (non)isomorphism are invariant-based: that is, they compute functions (invariants) that have the same value for isomorphic graphs and different values for nonisomorphic ones. For nonisomorphic input graphs G_1 and G_2, it usually matters what are their classes under permutations of their vertices and not which members of the classes are chosen (i.e., changing the order of vertices in G_1 hardly helps such an algorithm). This gives a motivation to the following relaxed notion of the

optimality. For the pair of graphs (G_1, G_2), call its *cluster* the set of pairs $(\pi_1(G_1), \pi_2(G_2))$ for all possible pairs of permutations (π_1, π_2) of their vertices. An acceptor A is called *optimal up to permutations* if for every algorithm B and every instance (G_1, G_2), the algorithm A runs in time polynomial in the size of the input and the maximum possible running time of the algorithm B on the cluster of (G_1, G_2).

We construct a randomized acceptor that is optimal up to permutations. Namely, we construct an algorithm that is always correct on nonisomorphic graphs, has a bounded probability of error on isomorphic graphs, and has the best possible median running time on every cluster. Moreover, our acceptor remains optimal if we strengthen the maximum running time on the cluster to the median running time on the cluster or even replace it by the order statistics $\min\{t \mid \Pr[\text{running time} \le t] \ge p\}$ for a constant p. This permits to reformulate the result in terms of the average-case complexity.

Average-case optimality. The basic notions of the average-case complexity were formulated by Levin [Lev73]. For a probability distribution D on the inputs, an algorithm is called average-case polynomial-time if there exists k such that the expectation of the k-th root of the running time is at most linear.

We introduce the notion of an average-case optimal algorithm: this is an algorithm that is as fast on average as any other algorithm for the same problem. The formal definition mimics Levin's definition of an average-case polynomial-time algorithm; in particular, if a problem can be solved in an average-case polynomial time, then the average-case optimal algorithm is an average-case polynomial one. Note that this notion lies in between of the conventional (pointwise) optimality and the worst-case optimality (an algorithm is worst-case optimal if it is polynomial-time if a polynomial-time algorithm for the same problem exists). For graph nonisomorphism, our optimality under permutations is a particular case of the average-case optimality for every distribution D that is stable under permutations of vertices (i.e., $D((G_1, G_2)) = D((\pi_1(G_1), \pi_2(G_2)))$ for every pair of nonisomorphic graphs (G_1, G_2) and permutations of their vertices (π_1, π_2)). Therefore, the acceptor that we constructed is average-case optimal with respect to every such distribution.

Organization of the paper. In Section 2 we introduce basic notation and definitions. Section 3 contains the proof of the main result: the construction of the algorithm that is optimal up to permutations. In Section 4 we introduce the notion of optimality on the average and conclude that our algorithm is optimal on the average for natural distributions. Section 5 contains open questions.

2 Definitions

2.1 Basic notation

The language `GNI` consists of all pairs of nonisomorphic undirected graphs with the same number of vertices.

For a graph G with n vertices and for a permutation $\pi \in S_n$, the graph $\pi(G)$ is the result of the application of π to the vertices of G.

An *ensemble D of probability distributions* is a sequence $\{D_n\}_{n\in\mathbb{N}}$ where each D_n is a probability distribution with finite support. We assume that D_n is concentrated on the inputs of length polynomial in n.

$U(S)$ denotes the uniform distribution on the set S.

For a random variable $X \in \mathbb{R}$, and for $p \in (0;1]$, we define $\mu^{(p)}[X] = \min\{t \mid \Pr[X \le t] \ge p\}$. In particular, $\mu^{(\frac{1}{2})}[X]$ is the median of X. If the probability space is not clear from the context, we denote it by a subscript; in particular, the internal random coins used by A are denoted by the letter A itself. For example, $\mu^{(p)}_{y \leftarrow D;A}$ means that y is taken according to the probability distribution D and A also uses its independent uniformly distributed random coins. Similarly, we use subscripts in the notations for the probability Pr and the expectation E. By default, they are taken over all random coins of algorithms under it.

For a randomized algorithm A, the random variable $\tau_A(x)$ denotes the running time of the algorithm A on input x. We denote by $t_A(x)$ the median of $\tau_A(x)$, i.e., $t_A(x) = \mu_A^{(\frac{1}{2})}[\tau_A(x)]$.

2.2 Randomized acceptors for graph nonisomorphism

Definition 2.1. A *randomized acceptor* for a language L is a randomized algorithm A such that

- for every input x, the algorithm A either accepts (written $A(x) = 1$) or does not stop (written $A(x) = \infty$);
- for every $x \in L$, $\Pr_A[A(x) = 1] > \frac{9}{10}$;
- for every $x \notin L$, $\Pr_A[A(x) = 1] < \frac{1}{10}$.

In what follows we will write "acceptor" instead of "randomized acceptor" for brevity.

Definition 2.2. Let $x = (G_1, G_2) \in \texttt{GNI}$. A *cluster* of x is the set $C_x = \{(\pi_1(G_1), \pi_2(G_2)) \mid \pi_1, \pi_2 \in S_n\}$.

We now introduce a notion of "simulations up to permutations" that is specific to the graph nonisomorphism problem.

Definition 2.3. An acceptor A for $\texttt{GNI}$ *simulates* an acceptor B, if there is a polynomial p such that for every $x \in \texttt{GNI}$,

$$t_A(x) \le p\left(|x| \cdot \mu^{(\frac{1}{4})}_{y \leftarrow U(C_x);B}[\tau_B(y)]\right).$$

Definition 2.4. An acceptor A for a language L is *polynomially bounded*, if there is a polynomial q such that for every $x \in L$, $t_A(x) \le q(|x|)$.

The following proposition follows from the definitions.

Proposition 2.1. Let A_1 and A_2 be two acceptors for $\texttt{GNI}$. If A_1 simulates A_2, and A_2 is polynomially bounded, then A_1 is polynomially bounded.

3 The result

Let A be a randomized algorithm, (G_0, G_1) be a pair of graphs, and δ be a rational number. The following procedure Amplify amplifies the probability of success for randomized acceptors.

Algorithm 3.1. Amplify(A, δ, G_0, G_1)

- Execute $m = \max\left\{\lceil 50 \ln \frac{2}{\delta} \rceil, 200 \ln 8\right\} + 1$ times in parallel:
  - Generate random permutations $\pi_0, \pi_1 \in S_n$;
  - Execute $A(\pi_0(G_0), \pi_1(G_1))$.
- Wait until $m/5$ of the executions stops, then stop and return 1.

When we say that we execute several algorithms *in parallel*, we mean that we simulate parallel executions sequentially. Namely, we execute one step of the first algorithm, then one step of the second algorithm, etc. until the last algorithm, then the next step of the first algorithm and so on. We need to run some algorithms this way, because we have no a priori bound on their running time and some of the executed algorithms even may not stop.

Lemma 3.1. Let A be a correct randomized acceptor for GNI.

1. If G_0 and G_1 be are isomorphic graphs, then $\Pr[\text{Amplify}(A, \delta, G_0, G_1) = 1] < \delta$.
2. There exists a polynomial p such that for every $x \in \text{GNI}$ the following inequality holds
$$\mu^{(1/4)}_{y \leftarrow U(C_x), \text{Amplify}}[\tau_{\text{Amplify}}(A, \delta, y)] \le p(\mu^{(1/4)}_{y \leftarrow U(C_x), A}[\tau_A(y)]).$$
3. For every $x \in \text{GNI}$, $\Pr[\text{Amplify}(A, \delta, x) = 1] > 1 - \frac{1}{20}$.

Proof. 1. Note that $\Pr_{\pi_0, \pi_1, A}[A(\pi_0(G_0), \pi_1(G_1))] < 1/10$. Then by Chernoff bounds $\Pr[\text{Amplify}(A, \delta, G_0, G_1) = 1] < 2e^{-\frac{2m}{100}} < \delta$.

2. Since Amplify applies a random permutation to the vertices of G_0 and G_1, then for every pair x of nonisomorphic graphs,
$$\mu^{(1/4)}_{y \leftarrow U(C_x), \text{Amplify}}[\tau_{\text{Amplify}}(A, \delta, y)] = \mu^{(1/4)}_{\text{Amplify}}[\tau_{\text{Amplify}}(A, \delta, x)].$$
By Chernoff bounds $n/5$ copies of A stop in $\mu^{(1/4)}_{y \leftarrow U(C_x), A}[\tau_A(y)]$ time with probability at least $1 - 2e^{-\frac{m}{200}} \ge \frac{1}{4}$.

3. By Chernoff bounds for every $x \in \text{GNI}$, $\Pr[\text{Amplify}(A, \delta, x) = 1] > 1 - 2e^{-0.98m} > 1 - \frac{1}{20}$. □

Let A be a randomized algorithm, (G_0, G_1) be a pair of graphs, and k be a number. The following procedure SelfCorrect employs A for checking that G_0 is not isomorphic to G_1 and verifies its answer.

Algorithm 3.2. SelfCorrect$(A, (G_0, G_1), k)$.

- If (G_0, G_1) is not a pair of graphs with the same number of vertices, diverge (go into an infinite cycle). Otherwise, assume n is the number of vertices in each of these graphs.
- Run k copies of Amplify$(A, (G_0, G_0), \frac{1}{20k})$ and one copy of Amplify$(A, (G_0, G_1), \frac{1}{20k})$ in parallel. Stop as soon as the first instance of Amplify accepts. If it was the instance that was given both G_0 and G_1, then accept; otherwise, diverge. (In case of a tie, diverge.)

The following algorithm amplifies the probability that SelfCorrect outputs its answer soon.

Algorithm 3.3. MultipleSelfCorrect$(A, (G_0, G_1), k, \ell)$.

- Run ℓ instances SelfCorrect$(A, (G_0, G_1), k)$ in parallel. Accept as soon as one of the instances accepts.

We now estimate the resulting probability of error.

Lemma 3.2. 1. For every pair $(G_0, G_1) \notin$ GNI, $\Pr[\text{SelfCorrect}(A, (G_0, G_1), k) = 1] \leq \frac{1}{k+1}$ and $\Pr[\text{MultipleSelfCorrect}(A, (G_0, G_1), k, \ell) = 1] \leq \frac{\ell}{k+1}$.

2. If A is an acceptor for GNI, MultipleSelfCorrect$(A, x, 30n, 3)$ is also an acceptor for GNI.

Proof. 1. If G_0 is isomorphic to G_1, then since Amplify randomly permutate its input all executions of Amplify are statistically identical.

2. Item 1 estimates the error probability on isomorphic instances. The probability of error on $x \in$ GNI can be bounded by the sum of the probability that Amplify$(A, (G_0, G_1), \frac{1}{20k})$ does not stop and k times the probability that Amplify$(A, (G_0, G_0), \frac{1}{20k})$ stops. The first term is less than $\frac{1}{20}$ and the second term is at most $k \times \frac{1}{20k}$ by Lemma 3.1. □

Lemma 3.3. Let A be a randomized acceptor for GNI. Then MultipleSelfCorrect$(A, x, 30n, 3)$ simulates A.

Proof. Consider the execution Amplify$(A, \frac{1}{2k}, (G_0, G_1))$ by SelfCorrect, where j is as defined in SelfCorrect. A single such execution stops accepting y_j in time $T = \mu^{(\delta)}_{y \leftarrow U(C_x);A}[\tau_{\text{Amplify}}(A, \frac{1}{2k}, (G_0, G_1))]$ with probability at least δ. Since different executions of SelfCorrect execute A on the inputs independently and uniformly chosen from the same cluster, the probability that A accepts the input in time T in at least one of these runs, is at least $1 - (1 - \frac{1}{3})^3 > \frac{1}{2}$. Clearly, the number of steps made by MultipleSelfCorrect is bounded by a polynomial of T, n. Finally, we apply Lemma 3.1(1). □

We now describe an acceptor Opt that simulates any other acceptor for GNI.

Algorithm 3.4. Opt(x).

- Let n be the number of vertices in one of the input graphs. Let A_i be the algorithm with Goedel number i, and let A_0 be the exponential-time deterministic acceptor for GNI that uses a brute-force search.
- For every $i \in \{0, 1, 2, \ldots, n\}$, run MultipleSelfCorrect$(A_i, x, 30n, 3)$ in parallel.
- Accept as soon as one of the parallel threads accepts.

Lemma 3.4. Opt is a randomized acceptor for GNI.

Proof. For $x \in$ GNI, the algorithm Opt(x) accepts because A_0 accepts. For $x \notin$ GNI, we estimate the probability $\Pr[\text{Opt}(x) = 1]$ as the sum of the probabilities of acceptance for the executions MultipleSelfCorrect$(A_i, x, 30n, 3)$. By Lemma 3.2 the resulting probability is less than $\frac{1}{10}$. □

Theorem 3.1. The algorithm Opt simulates every randomized acceptor for GNI.

Proof. Let B be a randomized acceptor for GNI. Then $B = A_i$ for some i. (Note that i does not depend on the input size n.) For $n < i$, the simulation holds because of the additive constant in a polynomial in the definition of the simulation. For $n \geq i$, the algorithm MultipleSelfCorrect will be applied to A_i; by Lemma 3.3 this execution simulates A_i. The fact that Opt runs only $(n + 1)$ algorithms in parallel, increases the running time only by a polynomial.

□

4 Optimality on the average

In this section we introduce the notion of optimality on the average and show that our algorithm constructed in Section 3 is optimal on the average for distributions that are invariant under permutations.

Definition 4.1 (cf. [BT06, Definition 2.4])**.** An algorithm A is *average-case polynomial-time* (or *polynomial-time on the average*) for an ensemble D, if there is $\epsilon > 0$ such that for every n,

$$\mathop{\mathrm{E}}_{x \leftarrow D_n} [t_A^\epsilon(x)] = O(n).$$

Remark 4.1. The difference from the definition in [BT06] is in the use of the median running time instead of the worst-case running time.

In what follows, we use this definition for distributions concentrated on pairs of nonisomorphic graphs; however, it does not affect the definition or the basic facts below. It also does not weaken the result: since we do not require anything (in particular, efficient samplability) from the distribution, one can use the result for any distribution restricted to nonisomorphic instances.

Definition 4.2. An acceptor A for a language L *simulates* an acceptor B *on the average* with respect to an ensemble D, if for every $\epsilon > 0$ there exists $c > 0$ such that for every $n \in \mathbb{N}$,

$$\mathop{\mathrm{E}}_{x \leftarrow D_n} t_A^c(x) = O(n \cdot \mathop{\mathrm{E}}_{y \leftarrow D_n} [t_B^\epsilon(y)]).$$

An acceptor that simulates on the average every other acceptor for L, is called *optimal on the average* w.r.t. D.

Remark 4.2. In a parallel work [HINS11], we define simulations on the average in a different way: while Definition 4.2 is designed so that Levin-style Definition 4.1 of average-case polynomiality is closed under these simulations, the definitions of [HINS11] are intended to work with Impagliazzo-style average-case polynomiality [Imp95].

The following proposition follows from the definitions.

Proposition 4.2. Let A_1 and A_2 be two acceptors for a language L, and let A simulate B on the average w.r.t. D. Then if B is average-case polynomial-time w.r.t. D, then A is also average-case polynomial-time w.r.t. D.

We now define a particular case of probability distributions that we use for the graph nonisomorphism problem.

Definition 4.3. An ensemble D of probability distributions is *natural*, if

- for every n, the support $\operatorname{supp} D_n$ contains pairs of nonisomorphic graphs on n vertices;
- D_n does not depend on the choice of a particular member of a cluster, i.e., for every pair $x \in \texttt{GNI}$ of graphs with n vertices and every $y \in C_x$, $D_n(x) = D_n(y)$. In other words, D_n is a convex combination of the uniform distributions on clusters.

Proposition 4.3. Let D be a natural ensemble. Assume that an acceptor A simulates an acceptor B. Then A simulates B on the average w.r.t. D.

Proof. We use the following lemma.

Lemma 4.1. Assume an acceptor A simulates an acceptor B. The for every $\epsilon > 0$ there exists $c > 0$ such that for every $x \in \mathtt{GNI}$, $t_A^c(x) = O(n \cdot \mathrm{E}_{y \leftarrow U(C_x)}[t_B^\epsilon(y)])$.

Proof. Denote $h = \mathrm{E}_{y \leftarrow U(C_x)}[t_B^\epsilon(y)]$. Markov's inequality implies that $\Pr_{y \leftarrow U(C_x)}[t_B^\epsilon(y) \le 2h] \ge \frac{1}{2}$. This inequality can be rewritten as $\Pr_{y \leftarrow U(C_x)}[\Pr[\tau_B^\epsilon(y) \le 2h] \ge \frac{1}{2}] \ge \frac{1}{2}$. Therefore, $\Pr_{y \leftarrow U(C_x);B}[\tau_B(y) \le (2h)^{1/\epsilon}] \ge \frac{1}{4}$, thus $\mu^{(\frac{1}{4})}_{y \leftarrow U(C_x);B}[\tau_B(y)] \le (2h)^{1/\epsilon}$. Since A simulates B, there exist a constant r such that $t_A(x) \le r(4n(2h)^{1/\epsilon})^r$. Assuming $c = \min\{\epsilon/r, 1/r\}$, we obtain the claimed asymptotics. □

Since a natural distribution is a linear combination of the uniform distributions on clusters, the lemma implies the proposition by the linearity of expectation. □

Corollary 4.1. Consider an acceptor O for $\mathtt{GNI}$ that simulates any other acceptor for $\mathtt{GNI}$. Then O is optimal on the average with respect to any natural ensemble of probability distributions.

Corollary 4.1 and Theorem 3.1 now imply the following statement.

Theorem 4.1. Opt is an acceptor for $\mathtt{GNI}$ that is optimal on the average with respect to any natural ensemble of probability distributions.

5 Conclusion and open questions

Our results may be generalized in the following way. Consider a sequence of sets M_n and groups H_n such that elements of M_n and H_n can be represented by words of size polynomial in n, and uniformly distributed elements of H_n can be sampled in a polynomial time. (The main result of this paper is for the case where M_n is the set of all labeled graphs with n vertices and H_n is the symmetric group of degree n.) Assume that for every n, the group H_n acts on the set M_n, and this action can be computed in a polynomial time. (For $\mathtt{GNI}$, an element of the group permutes the vertices of a graph.) Define the language

$$L = \bigcup_{n \in \mathbb{N}} \{(a, b) \mid a, b \in M_n, Orb(a) \cap Orb(b) = \emptyset\},$$

where $Orb(x)$ is the orbit of $x \in M_n$. (In this paper $L = \mathtt{GNI}$.) Define a cluster as the cartesian product of two orbits.

The main result of this paper can be easily generalized to the following statement: there exists a randomized acceptor for L that is optimal up to the induced action of H_n on L.

Open questions:

1. Generalize the result to "unnatural" ensembles of probability distributions.
2. Devise an average-case optimal randomized (possibly heuristic, see [HIMS11]) proof system for graph nonisomorphism.

Acknowledgements

The authors are grateful to anonymous referees for comments that improved the readability of the paper.

The first author is grateful to Olaf Beyesdorff, Nicola Galesi, and Massimo Lauria for discussions on the Arthur-Merlin protocol for graph nonisomorphism in Rome in December 2010.

References

[BK79] Laszlo Babai and Ludik Kucera. Canonical labelling of graphs in linear average time. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science (FOCS 1979)*, pages 39–46, 1979.

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundation and Trends in Theoretical Computer Science*, 2(1):1–106, 2006.

[CFM11] Yijia Chen, Joerg Flum, and Moritz Muller. Hard instances of algorithms and proof systems. Technical Report 11-085, ECCC, 2011.

[CK07] Stephen A. Cook and Jan Krajíček. Consequences of the provability of $NP \subseteq P/poly$. *The Journal of Symbolic Logic*, 72(4):1353–1371, 2007.

[HIMS11] Edward A. Hirsch, Dmitry Itsykson, Ivan Monakhov, and Alexander Smal. On optimal heuristic randomized semidecision procedures, with applications to proof complexity and cryptography. *Theory of Computing Systems*, 2011. To appear. Extended abstract appeared in the proceedings of *STACS-2010*. Preliminary version is available as ECCC TR10-193.

[HINS11] Edward A. Hirsch, Dmitry Itsykson, Valeria Nikolaenko, and Alexander Smal. Optimal heuristic algorithms for the image of an injective function. Technical Report 11-091, ECCC, 2011. Journal version is to appear in *Zapiski nauchnykh seminarov POMI* (*Notes of Mathematical Seminars of PDMI*).

[Hir10] Edward A. Hirsch. Optimal acceptors and optimal proof systems. In Jan Kratochvíl, Angsheng Li, Jirí Fiala, and Petr Kolman, editors, *TAMC*, volume 6108 of *Lecture Notes in Computer Science*, pages 28–39. Springer, 2010.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT'95)*, pages 134–147, 1995.

[KP89] Jan Krajíček and Pavel Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54(3):1063–1079, September 1989.

[Lev73] Leonid A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9:265–266, 1973. In Russian. English translation in: B.A.Trakhtenbrot. A Survey of Russian Approaches to Perebor (Brute-force Search) Algorithms. *Annals of the History of Computing* 6(4):384-400, 1984.

[Mes99] Jochen Messner. On optimal algorithms and optimal proof systems. In *Proceedings of STACS-99*, volume 1563 of *Lecture Notes in Computer Science*, pages 361–372, 1999.

[Mes00] Jochen Messner. *On the simulation order of proof systems.* PhD thesis, University of Erlangen, 2000.

[Sad99] Zenon Sadowski. On an optimal deterministic algorithm for SAT. In *Proceedings of CSL'98*, volume 1584 of *Lecture Notes in Computer Science*, pages 179–187. Springer, 1999.

[Sch76] Claus-Peter Schnorr. Optimal algorithms for self-reducible problems. In *ICALP'76*, pages 322–337, 1976.